The STAR Trigger Network

F. Bieser, H.J. Crawford, J.M.Engelage, E.G. Judd, J.M. Nelson, C. Perkins

22nd October 2004

1. **Hardware Configuration**

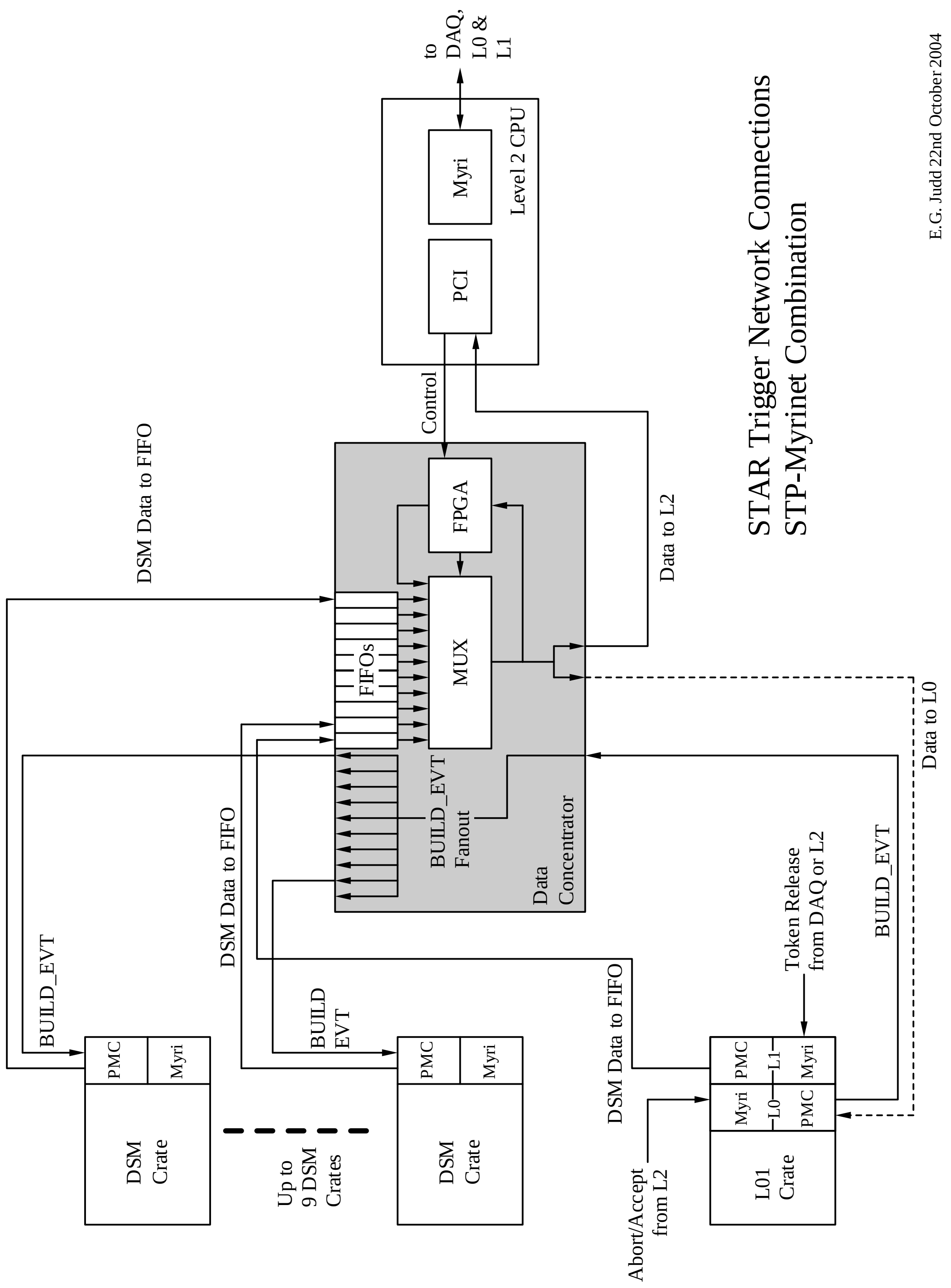

Figure 1: Hardware Layout of the STAR Trigger Network.

The new STAR Trigger Network will consist of a combination of Myrinet and the new STAR Trigger Data Pusher (STP) network hardware. The STP hardware consists of 3 separate modules:

- PMC Card
- Data Concentrator
- PCI Card

It is planned that every VME CPU in the trigger system will have both its current Myrinet card and a new PMC card. The PMC cards will all be connected to the Data Concentrator by fiber optic links. The Data Concentrator will then be connected to the PCI card in the Level 2 Linux computer (L2), again by a fiber optic link. L2 will have that STP PCI card and its current Myrinet card. Figure 1 shows a schematic diagram of how these modules will be connected. The BUILD_EVT command will be distributed from L0 to all the other VME CPUs via the Data Concentrator using STP. Trigger Data will be passed from all the VME CPUs to the Data Concentrator using STP and it will then pass the data on to L2, again using STP. Released tokens, abort and accept commands will be passed back to L0 and L1 using the existing Myrinet network. Communication between various VME CPUs that is not involved in the basic data flow (debug features, occasional scaler readout, etc…) will also continue to use the Myrinet network as it does now.

Flow control in the system will be managed by L1 in the same way as it does now. From the configuration information generated when a run is started L1 can determine the size of the smallest buffer in the system. It can then throttle the rate at which tokens are loaded into the TCU in order to ensure that this buffer never overflows and that there is always time for the DSM memories to be read out before they are over-written.

Control and status registers (CSRs) on the PMC and PCI cards will be accessible from their controlling CPU (VME or L2). Those registers on the Data Concentrator will be accessed by L2 through the PCI card.

1.1. PMC Card

The PMC cards will be installed in one of the two PCI Mezzanine Card slots in the STAR Trigger VME CPUs. Each PMC card therefore has a PCI connector. It also has an FPGA and 2 fiber optic connections; one for transmitting data and the other for receiving data. A schematic diagram of this card is shown in Fig. 2.

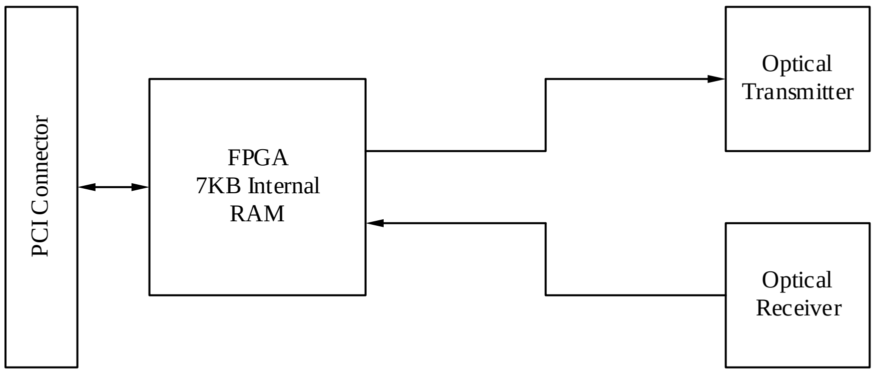

Figure 2: Schematic Diagram of the STP PMC Card

The card is controlled by the FPGA, which is responsible for:

- o Storing each packet that is received over the fiber optic link in an internal FIFO until it can be read by the CPU
- o Communicating with the CPU by interrupts over the PCI Bus so it is informed when a packet has arrived and can read it out
- o Receiving packets from the CPU over the PCI bus and then transmitting them out on the fiber optic link.

The FPGA has a 7KB internal block of RAM that will be split between one FIFO for storing the incoming packets from the fiber receiver and two more FIFOs for buffering the outgoing data before it is sent to the fiber transmitter.

1.2. Data Concentrator

The Data Concentrator will be installed in a separate 1U chassis on the platform. It has a dedicated input channel to receive the BUILD_EVT command. Data that comes in on this channel is just fanned out 10 times and sent out on 10 dedicated output channels. This is just a straight-through path; it does not involve any optical receivers or transmitters so there is no need for any control logic to resynchronize an optical receiver. There is therefore no intelligence in this path to intercept incoming commands, monitor them and modify or block the output; this path is uninterruptible and continuous.

The Data Concentrator also has 10 dedicated input channels for receiving data packets from up to 10 DSM crates. Each channel has a 128KB FIFO for storing that data. A state machine implemented in an FPGA will survey these FIFOs continuously in a round-robin fashion. The survey will include an 11th FIFO implemented inside the FPGA itself that will contain status information from the FPGA. When it encounters a FIFO that is not empty the FPGA will read data from that FIFO, and pass each word on to the PCI card in L2, until it has transferred a full packet. Only then will it move on to the next FIFO. As an alternative (for debugging purposes or backup) there is a second copy of the output that is sent to L0. Under normal running conditions L0 will be configured to ignore data received from this link.

Finally, the Data Concentrator has a fiber input connector from the PCI card in L2 to enable L2 to send commands to the FPGA. Any data that is produced as a result of those commands is stored in the 11th FIFO and read out in turn.

1.3. PCI Card

The L2 PCI card is physically almost the same as the PMC card, but it has an additional 128KB FIFO inserted between its optical receiver and the FPGA. A schematic diagram of the PCI card is shown in Fig. 3.

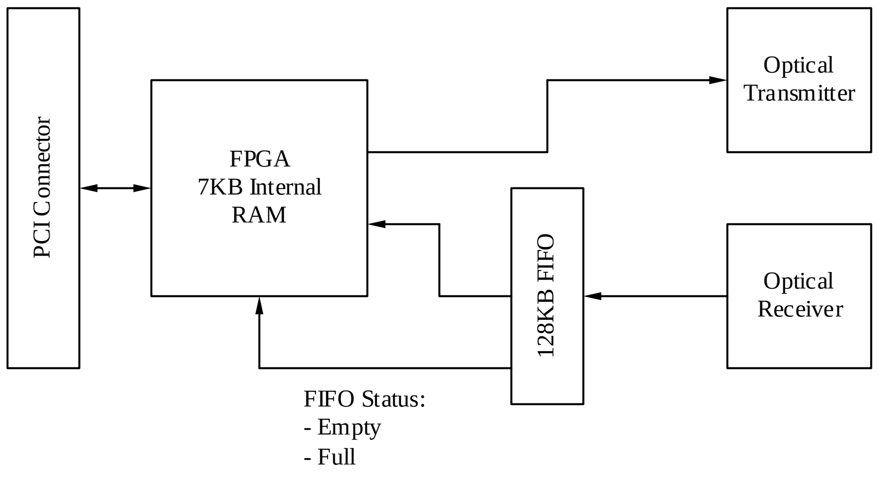

Figure 3: Schematic Diagram of the STP PCI Card

This FPGA is responsible for:

- Transferring full data packets from the FIFO to its internal 7KB RAM
- Transferring full data packets from its 7KB internal RAM to the next available buffer in L2 using a DMA transfer
- Interrupting L2 to notify it when a DMA transfer has been completed..
- Sending commands to the Data Concentrator to configure and monitor its status.

2. Control and Status Registers

2.1. PMC

2.1.1. CSR0 – Reset and General Status

The PMC card needs to have a register that will contain basic hardware information. For example there would be a bit to indicate if the Optical Receiver had ever detected any link errors and three bits to indicate if any of the three FIFO-full bits was ever set. A write to this register from the VME CPU should be used to reset the PMC card to its clean, power-on condition: the Optical Receiver should be resynchronized and all three of the internal FIFOs should be cleared, which should result in all the status bits (see also CSR6) being reset.

2.1.2. CSR1 – Interrupt Enable

The PMC card needs to have a 2-bit register to configure the card for its running mode. 1 bit will be used to enable or disable the generation of interrupts on the PCI bus. The default state of this register on power-up will be to disable the generation of interrupts. Any data packets that arrive when the PMC card is in this state will essentially be ignored. When Run Control sends a START_RUN command to the CPU it will then modify the register on the PMC card to enable the generation of interrupts. From then on when a BUILD_EVT command arrives it will be stored locally on the PMC card, which will then raise an interrupt on the PCI Bus to inform the CPU. When the CPU receives the STOP_RUN command from Run Control it will disable the generation of interrupts. If a CPU is not included in the run then it will never receive any commands from Run Control, the PMC card will be left in the state with interrupts disabled and the CPU will therefore never receive spurious BUILD_EVT commands or send unwanted data to the Data Concentrator. The second bit will be used to zero the incoming and outgoing packet counters. It must be possible for the VME CPU to read the current status of this register to monitor whether the PMC card has interrupts enabled or not.
NOTE: Under normal running conditions interrupts will never be enabled in the L0 CPU. That CPU generates the BUILD_EVT command, it does not receive it. The fiber input to L0 comes from the backup output of the Data Concentrator, which is not normally used.

2.1.3. CSR2 – Small Outgoing Packet FIFO, Word Access

The PMC card needs to implement a 128x32 FIFO (i.e. 512 bytes). The FIFO should be 32 bits wide since that is the width of the PCI bus. The FIFO should be 128 words deep since that is large enough to hold multiple BUILD_EVT commands. When a VME CPU needs to transmit a data packet that is so small it would be inefficient to use a DMA transfer then the CPU will just write each word of the data packet into this FIFO, one word at a time. If the FIFO-full bit is ever set this should be considered an error condition, and the relevant bit in CSR0 should be latched on.

2.1.4. CSR3 – Large Outgoing Packet FIFO, DMA Access

The PMC card also needs to implement a 1400x32 FIFO (i.e. 5.6 Kbytes). The FIFO should be 32 bits wide since that is the width of the PCI bus. The FIFO should be 1400 words deep since that is larger than the largest DSM data packet (BC1: 4 words/DSM, 17 DSMs, 11 buffers = 748 bytes) and also uses up the remainder of the 7 KB internal RAM. When a VME CPU needs to transmit a data packet that is large enough to make DMA use efficient then it will write that packet into this FIFO using a DMA transfer. If the FIFO-full bit is ever set this should be considered an error condition, and the relevant bit in CSR0 should be latched on

2.1.5. CSR4 – Outgoing Packet Control

The PMC card needs to have a 4-bit write-only CSR to allow the VME CPU to notify the PMC card of the beginning and end of data packets that are to be transmitted. 2 bits will be used for the small word-access FIFO (CSR2) and the other 2 bits are for the large DMA-access FIFO (CSR3). When the VME CPU is about to start sending a packet to the PMC card it will first write a “1” to the relevant 2-bit field of this register. That will prompt the FPGA to generate a command marker, indicating “beginning of packet”, and send it out on the Optical Transmitter. After the VME CPU

has finished sending all the data in this packet to the PMC card it will then write a “2” to the relevant 2-bit field of this register, which will prompt the FPGA to generate a second command marker indicating “end of packet” and send that word out to the Optical Transmitter.
NOTE: The optical transmit and receive chips actually exchange 18 bits; 16 data bits and 2 control bits. A “command” marker is one in which the 17th bit (i.e. one of the 2 control bits) is set to “1”. If that bit is set to “0” then it is treated as a “data” word. On the receiving end of the optical link this flag can be used to distinguish command markers from data words. The second control bit is a “range” flag. When this bit is set the data should be treated as the upper 16 bits of a 32-bit word.

2.1.6. CSR5 – Incoming Packet FIFO

The PMC card needs to implement a 128x32 FIFO (i.e. 512 bytes). The FIFO should be 32 bits wide since that is the width of the PCI bus. The FIFO should be 128 words deep since that is large enough to hold multiple BUILD_EVT commands. Any data that is received from the Optical Receiver should be written into this FIFO. When at least one full packet has been stored in the FIFO the VME CPU will be notified and the CPU can then read the packet data from this register. Data will be read one word at a time, i.e. no DMA. If the FIFO-full bit is ever set this should be considered an error condition, and the relevant bit in CSR0 should be latched on

2.1.7. CSR6 – FIFO Status

The PMC card needs to have a 1-bit read-only CSR to allow the VME CPU to determine if there are any more full packets to be read from the FIFO. The bit would indicate when the FIFO was “empty” (i.e. it contained no full packets) or “not empty” (i.e. it contained at least one full packet). When the first packet arrives from the Optical Receiver the FPGA writes it into the FIFO and uses the write of the “end of packet” command marker to increment a counter by 1. (NOTE: There may be no need to write the “beginning of packet” command marker into the FIFO, just start with the first data word). The change in the FIFO counter from zero to a non-zero value will change the status from “empty” to “not empty”. This status change will prompt the FPGA to generate an interrupt on the PCI bus to inform the VME CPU that a packet is available. If more packets are received before the VME CPU has read out the first one they are just stacked up in the FIFO, the FIFO counter is incremented for each one but there is no change to the FIFO Status bit. When the VME CPU receives the interrupt it reads 32-bit words from the FIFO (see CSR5 above). After reading each word it checks the status bit to see if the FIFO is now empty and if it is not then the CPU reads more words until the FIFO is empty. In parallel the FPGA will decrement the FIFO counter by 1 whenever the VME CPU has read out a full packet (as determined by reading the “end of packet” marker from the FIFO). If the counter value is now zero again the FIFO status is changed from “not empty” to “empty”. This will be implemented in such a way that the “beginning of packet” and “end of packet” markers do not themselves get read by the VME CPU.

2.1.8. CSR7 – Incoming Packet Counter

The PMC card needs to have a read-only 32-bit register that will hold the current value of a 32-bit counter. The counter will be reset to zero whenever a new run starts (interrupts are enabled, see CSR1) and will then be incremented by 1 whenever a full packet arrives over the optical link (as determined by receipt of the “end of packet” command marker).

2.1.9. CSR8 – Outgoing Packet Counter

The PMC card needs to have a read-only 32-bit register that will hold the current value of a 32-bit counter. The counter will be reset to zero whenever a new run starts (interrupts are enabled, see CSR1) and will then be incremented by 1 whenever a full packet has been transmitted over the optical link (as determined by generation of the “end of packet” command marker).

2.2. Data Concentrator

The Data Concentrator is an independent module installed in a chassis. It has no general Ethernet or Myrinet connection and no VME or PCI interface. It does have an input fiber optic connection from the PCI card on L2. It also has access to the 11th MUX input to allow it to send its own data to that PCI card. Any communication will therefore involve the PCI card sending a packet to the Data Concentrator that contains a command and, at some convenient time later, the Data Concentrator can form a data packet with any requested information and send that back to the PCI card.

2.2.1. CSR0 – Reset Hardware

When the FPGA receives a packet containing the RESET command from the PCI card it should reset the Data Concentrator to its clean, power-on condition: the 11 Optical Receivers (10 for the FIFOs and 1 for the PCI card connection) should be resynchronized and the 11 FIFOs (10 external and 1 internal) should be cleared, which should result in all the associated status bits being cleared.

2.2.2. CSR1 – Send Status

When the FPGA receives a packet containing the SEND_STATUS command from the PCI card it should form a standard packet, i.e. one that contains the “beginning of packet” command marker, several data words and then the “end of packet” command marker. One data word should contain 11 bits indicating which, if any, of the 11 Optical Receivers ever detected a link error. Another data word should contain 11 bits indicating which, if any, of the 11 FIFO-full bits were ever set. Since this packet is going to be sent to the PCI card and then on to L2 it should also contain data words with any header information needed by L2 (e.g. a source ID word to distinguish this data from DSM data). This data packet should be stored in the internal FIFO so it can be read out and sent to the PCI card in L2.

2.2.3. CSR2 – Reset Counters

The FPGA needs to implement 11 packet counters, one for each of the 11 FIFOs. When the FPGA receives a packet containing the RESET_COUNTERS command from the PCI card it should reset each of those 11 counters to zero. This command will be issued at the start of every run. After that each FIFO counter should be incremented by one whenever the FPGA determines that it has read a full packet from that FIFO and sent it to the PCI card (as determined by the transmission of the “end of packet” marker).

2.2.4. CSR3 – Send Counters

When the FPGA receives a packet containing the SEND_COUNTERS command from the PCI card it should form a standard packet, i.e. one that contains the “beginning of packet” command marker, several data words and then the “end of packet” command marker. The packet should contain 1 data word for each of the 11 counters to hold the current value of that counter. Since this packet is going to be sent to the PCI card and then on to L2 it should also contain data words with any header information needed by L2 (e.g. a source ID word to distinguish this data from DSM data). This data packet should be stored in the internal FIFO so it can be read out and send to the PCI card in L2.

2.3. PCI

2.3.1. CSR0 – DMA Address FIFO

L2 will load the PCI card DMA Address FIFO with up to 32 DMA addresses by writing each address value to this register. Each address will point to the start of a buffer in L2 that is large enough to contain the largest possible DSM data packet (~ 3KB). When the PCI card has received a data packet from the Data Concentrator it will check to see if there is a DMA address available (i.e. this FIFO is not empty) and if so it will read an address from this FIFO and then execute a DMA transfer to push the data packet from the PCI card into the buffer with this address in L2’s memory.

2.3.2. CSR1 – Used DMA Address FIFO

Once the PCI card has completed the DMA transfer of a data packet to L2 it will write the used DMA address into this FIFO before interrupting L2 to notify it that a DMA has been completed. L2 will then retrieve the address of the used buffer by reading it from this register.

NOTE: There are various schemes for dealing with the case where more packets arrive from the Data Concentrator while the DMA of the first packet is in progress. Chris will experiment with these during the development of the PCI card and see which works best.

2.3.3. CSR2 – DMA Words to Transfer (Debug Only)

In order to debug the DMA function it is necessary to be able to set up and initiate a DMA transfer. The DMA address is loaded into CSR0 in the normal way. L2 can specify the number of 32-bit words to be transferred by writing to this register.

2.3.4. CSR3 – Start DMA Transfer (Debug Only)

When L2 writes to this register the debug DMA transfer will actually be initiated.

2.3.5. CSR4 – Reset PCI Card and Data Concentrator

The PCI card needs to have a read/write register which will contain basic hardware information. For example there would be a bit to indicate if the Optical Receiver had ever detected any link errors and another bit to indicate if the FIFO-full bit is ever set. A write to this register from L2 should be used to reset the PCI card to its clean, power-on condition: the Optical Receiver should be resynchronized, the external FIFO and all the FIFOs built into the FPGA should be cleared, which should result in their status bits being cleared. Writing to this register should also prompt the PCI card to send a packet containing the RESET command to the Data Concentrator.

2.3.6. CSR5 – Generate Data Concentrator "Send Status" Command

The PCI card needs to have a write-only register to allow L2 to instruct the Data Concentrator to generate and transmit its status information. When L2 writes to this register the PCI card should send a packet to the Data Concentrator containing the SEND_STATUS command.

2.3.7. CSR6 – Reset Packet Counters

The PCI card needs to have a write-only register to allow L2 to notify it when a run is about to start so all packet-counters should be zeroed. When L2 writes to this register the PCI card should send a packet to the Data Concentrator containing the RESET_COUNTERS command. The PCI card also needs to reset its own, internal, incoming packet counter (see CSR7 below).

2.3.8. CSR7 – Incoming Packet Counter

The PCI card needs to have a read-only register with a large number of bits that will hold the current value of its incoming packet counter. The counter will be reset to zero whenever a new run starts (CSR6) and will then be incremented by 1 whenever a full packet has been read from the FIFO and transferred to L2 in a DMA transfer

2.3.9. CSR8 – Generate Data Concentrator "Send Counters" Command

The PCI card needs to have a write-only register to allow L2 to instruct the Data Concentrator to latch the current values of its packet counters and send the data to the PCI card. When L2 writes to this register the PCI card should send a packet to the Data Concentrator containing the SEND_COUNTERS command

3. Data Flow

The method by which a trigger would be issued in this scheme is:

- When Run Control sends the START_RUN command all the VME CPUs will enable interrupt generation on their PMC cards (except L0 and L1) and zero their packet counters. L2 will instruct the PCI card to reset its own incoming packet counter and send the "Reset Counters" command to the Data Concentrator.
- L0 is polling on the TCU, it sees a trigger has been issued, reads the INFO FIFOs and builds the event descriptor just like now.
- L0 (not L1) sends the BUILD_EVT command out to the Data Concentrator via the output of its PMC card. The Data Concentrator fans out that command and distributes a copy to the PMC card

in each the DSM crates. Each of these PMC cards notifies its CPU after the full packet has arrived. This takes very little time, unlike now. **NOTE**: The Data Concentrator does not need a BUILD_EVT connection to L1 since L1 can receive the command directly from L0, which is in the same crate. This will save on the interrupt handling time in L1.

- L0 sends the event descriptor and a BUILD_EVT command to L1 over Myrinet.
- L1 reads its DSM boards, executes its algorithm and maintains its counters as usual. If the algorithm is successful L1 sends its data, and the event descriptor, to the Data Concentrator. If the algorithm is unsuccessful L1 sends an ABORT command to L0 (to be loaded into the TCU). L1 also sends a data packet to the Data Concentrator that contains just an ABORT command instead of its usual DSM data.
- All other CPUs read their DSM boards when they receive the BUILD_EVT command from the Data Concentrator and send the data to the Data Concentrator via the output of their PMC card.
- The Data Concentrator is surveying each of its FIFOs and passes each data packet as it arrives to the PCI card in L2.
- The PCI card receives all the data packets from the Data Concentrator and passes each one to L2.
- L2 collects all data packets from all CPUs that are in the run for each event. If the data packet from L1 contains the ABORT command then L2 will discard all the data and release the token back to the Token Manager in L1 using Myrinet. If not L2 will execute its own algorithm as is done now.
- L2 then sends either its Abort or Accept back to the L0 CPU using Myrinet exactly as is done now. If the event is accepted it is sent on to DAQ via the usual Myrinet connection. DAQ can send released tokens directly back to L1 just as it does now over Myrinet.
- When L0 receives an ABORT command it will now register whether that command comes from L1 or L2. If the ABORT comes from L1 then L0 will load it into the TCU but then take no further action: L2 is responsible for releasing the token to the Token Manager for L1-generated ABORTS. However, if the ABORT comes from L2 itself, L0 will load it into the TCU and, once the ABORT has actually been issued to the detectors, L0 will release the token to the Token Manager.

4. Monitoring

At periodic intervals the general hardware status registers on each PMC card (CSR0), the Data Concentrator (CSR1) and the PCI card (CSR4) will be read out. If there are any indications of link errors or FIFO overflows then a log message should be sent to Run Control. If necessary a FORCE_RUN_STOP command can be issued.

5. Communications Protocol

The optical transmitters and receivers on the PMC cards and Data Concentrator move data as 16-bit words with 2 additional control bits one of which specifies if the word is a data word or a command word. Within STP we will use a common format for all packets sent over the optical links. The first and last word in any optical transmission will be command markers indicating the beginning and end of the packet. These command words will be used by the FPGA on the receiving end to determine when it has received a full packet:

Word	Type	Data
0	Command	Beginning of Packet Marker
1:N	Data	Packet Data
N+1	Command	End of Packet Marker

In order to minimize differences with the current Myrinet-only system the format of the packet data for the BUILD_EVT command will be the same as the format of the current Myrinet message buffer. Also the format of the packet data sent from each VME CPU to the Data Concentrator will be the same as is currently used in the MyriMemcpy2 calls, with the addition of the DSM crate identifier. Since neither of these 2 varieties of packets are actually unpacked and used by any of the STP modules (they just get

passed through) the details of those formats are left to the discretion of the VME CPU and L2 programmers.
NOTE: The packet format for commands sent from the PCI card to the Data Concentrator needs to be determined.

6. <u>PCI Bus Collisions</u>

There is also a separate issue of clashes between interrupts from the PMC card, the Myrinet card and the DMA engine (used to read out the DSM boards) on the PCI Bus. We have already seen that when the fast DMA code is used, which involves polling on the PCI Bus, while there is activity on Myrinet that the two can clash and generate bus errors.
QUESTION: Is the addition of another card that uses the PCI Bus going to make matters even worse?